Single-Encoder VAE for Image-To-Image Translation

Adit Gajjar **Haresh Nasit** **Filip Balucha**

Abstract

Image-to-image translation is a computer vision problem where an image is mapped from one domain to another. In the supervised scenario, the dataset consists of perfectly-paired image data. However, such data is either hard to find or simply does not exist [11]. Therefore, we focus on the unsupervised setting, where a joint distribution of images across two domains is learnt using the marginal distributions – the images in individual domains [5]. We propose a model that builds on the shared latent space assumption and uses domain-specific decoders and readily extends to more than two different domains. The implementation is available here.

1 Introduction

Image-to-image translation aims to capture the characteristics of one image collection and determine how these could be translated into another image collection [11]. Mathematically, the goal is to learn a mapping $G : X \to Y$, where X is the source and Y the target domain. Image-to-image translation includes many computer vision problems, such as super-resolution, colorization and style transfer [5].

Many efforts were put into the supervised setting, where models learn image-to-image translation across two domains using perfectly-paired images [11]. However, obtaining paired data is often difficult and expensive, and sometimes impossible [11]. For example, to learn the translation from photographs to images styled as Monet paintings, the supervised setting would require pairs of real-life images and the corresponding Monet paintings – an infeasible task. Therefore, we focus on unsupervised algorithms, which map between domains without paired examples.

Probabilistically, the aim is to learn a joint distribution of images between two different domains, where each domain consists of images from a marginal distribution [5]. However, there are infinitely many such joint distributions [5]. To address this problem, we can add more structure to the objective [11] or make assumptions about the structure of the joint distribution [5]. CycleGAN achieves this by training two mappings $G : X \to Y$ and $F : Y \to X$, and updating the objective to enforce $F(G(x)) \approx x$ and $G(F(y)) \approx y$ [11]. By contrast, Liu et al. [5] assume a shared latent space, where corresponding images from different domains are mapped to the same latent representation.

Inspired by the stable training and meaningful encodings characteristic of variational autoencoder (VAE) models, we continue in the latter vein. We propose a model that leverages the shared latent space assumption but is simpler than the model proposed by Liu et al. [5]. While our model does not outperform theirs, we propose future directions that could bring our architecture to fruition.

2 Related works

Image-to-image translation is a well-studied problem. We provide a summary of papers that tackle the unsupervised setting. We focus on two well-known architectures – the popular Generative Adversarial Networks (GANs) and the Variational Autoencoders (VAEs), praised for learning useful representations that lend themselves to manipulation.

GANs. CycleGAN builds on the successful GAN architecture by introducing cycle-consistency. This constrains the image-to-image translation problem and encourages meaningful translations [11].

VAEs. Several methods use VAEs in the context of image-to-image translation. ST-VAE projects styles to a linear latent space and merges them using linear interpolation [6]. Kazemi et al. [2] propose a framework that models separate representations for image content and style. Larsen et al. [4] propose VAE-GAN, which combines a VAE and the discriminator from the GAN architecture. Finally, Jha et al. [1] apply the cycle-consistency idea to VAEs.

VAEs and Shared latent representation. Several research efforts establish a shared latent space. Zhao and Chen [10] propose a mapping between the two latent spaces corresponding to the two domains. Perhaps closest to our approach, Liu et al. [5] share the latent space z between the two domains by sharing some weights of domain-specific encoders and decoders. Our simplified architecture achieves the same using a single, shared encoder.

3 Method

3.1 Objective

The objective adapts the vanilla VAE objective that consists of a variational and a reconstruction term.

3.1.1 Variational loss

The variational is the KL divergence between the approximated posterior and the prior distribution and thus regularizes the latent space. The posterior is the encoder distribution, and the prior captures which distribution the encoder distribution should be regularized to. Similar to Kingma and Welling [3], we assume both distributions are Gaussian, so that the variational loss term can be expressed as

$$\mathcal{L}_{\text{Variational}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \sum_{j=1}^{J} \left(1 + \log\left(\left(\sigma_j^{(i)}\right)^2\right) - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2\right) \tag{1}$$

where σ and μ determine the approximated encoder distribution, J is the latent dimension, and i ranges over all samples in the batch $\mathcal{B}$.

3.1.2 Reconstruction loss

Kingma and Welling [3] use a MSE reconstruction loss function [3]. However, in the context of images, this assumes that the impact of noise is independent of the local characteristics [9]. We therefore use a multiscale structural similarity index (MS-SSIM) term. MS-SSIM is a differentiable similarity metric that takes into account local features at various scale [9] and aims to produce visually-pleasing outputs. Using the MS-SSIM term alone led to slow convergence. Therefore, similar to Zhao et al. [9], we combine MSE and MS-SSIM loss. The reconstruction loss thus becomes

$$\mathcal{L}_{\text{Reconstruction}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \text{MSE}\left(\mathbf{u}^{(i)}, \tilde{\mathbf{u}}^{(i)}\right) + \text{MS-SSIM}\left(\mathbf{u}^{(i)}, \tilde{\mathbf{u}}^{(i)}\right) \tag{2}$$

3.1.3 Full loss

Finally, our full objective can be expresses as

$$\mathcal{L} = \mathcal{L}_{\text{Variational}} + \alpha \cdot \mathcal{L}_{\text{Reconstruction}} \tag{3}$$

where α is used to account for different learning rates between the two loss terms.

3.2 Architecture

Training 1: Encoder. Our architecture combines the components of the standard convolutional VAE (Fig 1). Both domains share a single encoder E, which is trained in concord with a shared decoder D. Its purpose is to establish a shared latent space where, based on the shared latent space assumption, images with similar content from different domains will be mapped to a similar point.

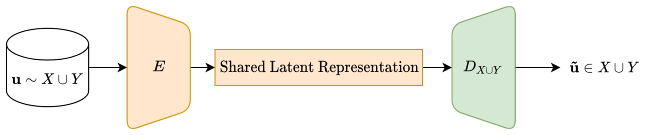
$\mathbf{u} \sim X \cup Y$
E
Shared Latent Representation
$D_{X \cup Y}$
$\tilde{\mathbf{u}} \in X \cup Y$

Figure 1: Learning a shared latent representation for both image domains.

Algorithm 1 The procedure for training the encoder E

```
// $X, Y \leftarrow$ loadDataset()
$M \leftarrow$ new VAE()
$E \leftarrow$ new Encoder()
$M$.encoder $\leftarrow E$
$D \leftarrow$ new Decoder()
$M$.decoder $\leftarrow D$
$M$.train($X \cup Y$)
```

Training 2: Domain-specific decoders. Once the encoder is trained its weights are frozen to fix the mapping to a shared latent space. Then, a separate decoder is trained for each domain (Fig 2). For example, when translating between images of apples and oranges, E would encode images of both, apples and oranges, and apples and oranges would each have a dedicated decoder.

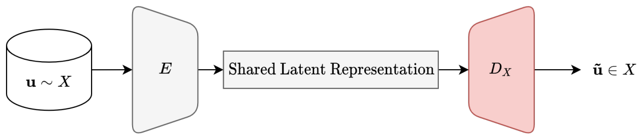

Figure 2: Learning the decoder D_X for domain X that uses embeddings from the shared latent space.

Style transfer. To perform style transfer, we combine the encoder and a domain-specific decoder. For example, to translate between domains Y and X, we use the encoder E and decoder D_X (Fig 3).

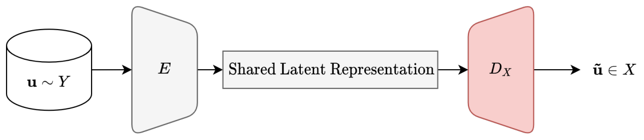

Figure 3: Translation from Y to X using shared encoder E and domain-specific decoder D_X.

4 Experiments

We performed image translation experiments using three different datasets. The hyperparameters used to train on all the datasets can be found in Table 1. The model results in translations that have some characteristics of the target domain, but the outputs still have blurry edges and lack clarity.

4.1 `apple2orange`

Our model learnt the shape and color of the fruits well. For instance, in the below translation of orange to apple, the translation removed the leaf and the stem of the orange. Qualitatively, the translations of orange to apple look better than vice versa (Fig 4).

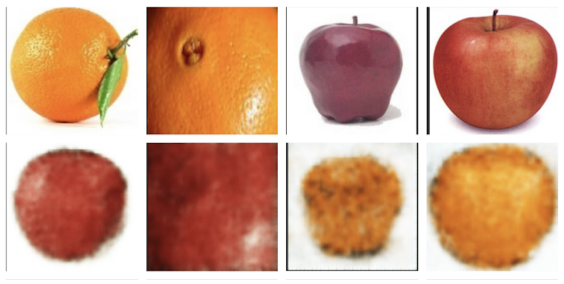

Figure 4: Image translations between the "apple" and "orange" domains. The first line shows the original image, the second the translated version.

4.2 `summer2winter_yosemite`

Our model's performance on `summer2winter_yosemite` was not as good as `apple2orange` dataset. Qualitatively, we can tell the difference between the translations of the two seasons. The model learnt the color of the sky and trees. The translations of summer images to winter resulted in images that had gray skies and black trees; vice versa, the outputs had blue skies and green trees (Fig 5).

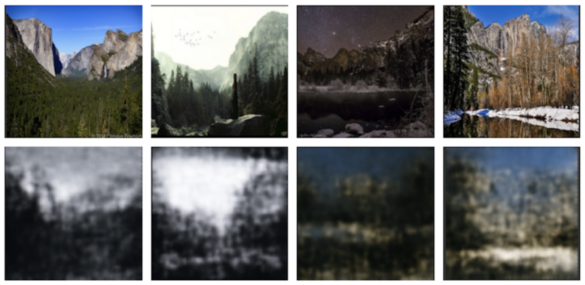

Figure 5: Image translations between the "winter" and "summer" domains. The first line shows the original image, the second the translated version.

4.3 `horse2zebra`

Our model performed the worst on the `horse2zebra` dataset. Qualitatively, it is impossible to tell the difference between the translations of the two animals as the translations only capture the silhouette. It is possible that the model was unable to learn the physical features of the two animals since the horse dataset consists of horses of multiple colors (Fig 6).

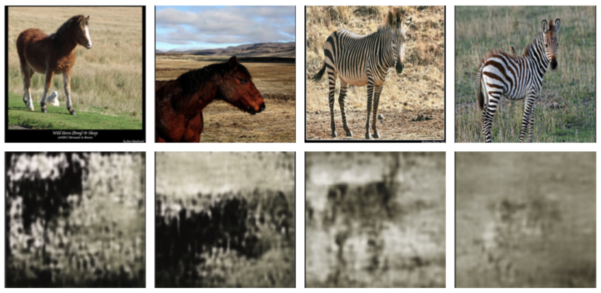

Figure 6: Image translations between the "horse" and "zebra" domains. The first line shows the original image, the second the translated version.

5 Conclusion and future work

Our VAE architecture performs worse than CycleGAN or the related architecture proposed by Liu et al. [5]. While the output content (e.g. the rough horse and zebra silhouettes) is maintained, the output quality is subpar. Our domain-specific decoder was likely limited by the quality of the latent representation produced by the shared encoder.

A more complex encoder, such as the one proposed by Liu et al. [5], could learn a better representation of the domains and thus produce better translations. Another way to produce higher-quality images would be to replace our standard VAE architecture with a hierarchical VAE or VQ-VAE which has been shown to produce higher quality reconstructions [8]. In addition, a more-informed loss function could improve quality. For instance, similar to Liu et al. [5], a loss term based on a discriminator network could be used for translated images.

References

[1] A. H. Jha, S. Anand, M. Singh, and V. S. R. Veeravasarapu. Disentangling factors of variation with cycle-consistent variational auto-encoders. Apr. 2018.

[2] H. Kazemi, S. Soleymani, F. Taherkhani, S. M. Iranmanesh, and N. M. Nasrabadi. Unsupervised image-to-image translation using domain-specific variational information bound. Nov. 2018.

[3] D. P. Kingma and M. Welling. Auto-Encoding variational bayes. Dec. 2013.

[4] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. Dec. 2015.

[5] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. Mar. 2017.

[6] Z.-S. Liu, V. Kalogeiton, and M.-P. Cani. Multiple style transfer via variational AutoEncoder. Oct. 2021.

[7] A. Subramanian. Pytorch-vae. `https://github.com/AntixK/PyTorch-VAE`, 2020.

[8] A. Vahdat and J. Kautz. NVAE: A deep hierarchical variational autoencoder. July 2020.

[9] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for neural networks for image processing. Nov. 2015.

[10] Y. Zhao and C. Chen. Unpaired image-to-image translation via latent energy transport. Dec. 2020.

[11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. Mar. 2017.

Appendix

A Training details

A.1 Layer dimensions

The shared encoder E and the domain-specific decoders D_x and D_y have the same size. Both the encoder and decoders are 5-layer deep with the following hidden sizes: 32, 64, 128, 256 and 512. The latent representation was a 256-dimensional vector. Each encoder and decoder layer included, in sequence, a convolution or transposed convolution, batch normalization, and the Leaky ReLU activation function. We use the Adam optimizer with a batch size of 64 and $\gamma = 0.95$. Hyperparameters used in the appendix.

A.2 Datasets

We trained the model on the `apple2orange`, `summer2winter_yosemite` and `horse2zebra` datasets. The images were split into domain-specific (i.e. apple and orange) datasets. The domain-specific datasets were used to train domain-specific decoders, and the combined domain dataset was used to train the shared encoder. The images used were of dimensions 256×256.

A.3 Training

The model was implemented by adapting code from Subramanian [7]. The model was trained on Google Colab using a single NVIDIA Tesla K10 GPU. The shared encoder and domain-specific decoders were trained for a maximum of 70 epochs, respectively. The training duration was approximately 80 minutes.

B Hyperparameters

Table 1: The choice of hyperparameters

Hyperparameter	Value
Training batch size	64
Validation batch size	64
Learning rate	0.0005
γ (Adam)	0.95
α ($\mathcal{L}$)	0.00025

C Individual contributions

The team collaborated closely across all phases of the project, including ideation, research, implementation, evaluation and writing. Adit experimented with hyperparameters and trained all the models that were used to generate the samples presented in the report. Haresh helped to implement the model. He contributed to the Abstract, Introduction, Training details, Experiments, and Conclusion sections of this report. Filip led the live coding sessions held by our group, created diagrams from sketches, referenced research in the Abstract, Introduction and Related works sections. He was also responsible for the bibliography.